Anomalies Analysis and Detection Using Computer Vision for Finding Defects in Plant Leaves Images

Bohdan Koval and Iulia Khlevna

Taras Shevchenko National University of Kyiv, Volodymyrska str., 60, Kyiv, 01033, Ukraine

Abstract

The work is dedicated to the research of anomalies detection in visual data using computer vision tools. It describes and combines existing deep learning algorithms to build a composite, adjusted computer vision model for plant leaves image analysis and finding defects on them. In this example, the work demonstrates how to apply existing foundational deep learning tools effectively, and tune and combine them per requirements to achieve high-level scoring on real-life tasks. The task is formulated as one of the types of multidimensional data classification problem. The basis is an applied study of detecting defects (anomalies) in images of plants, which represent a set of 1820 3-dimensional matrices that are a subject of further processing using deep learning algorithms, particularly computer vision. The paper demonstrates the appliance of the Canny algorithm to detect edges in the images to extract useful information (object of research). Also, the article gives examples of incoming data stream processing, pre-processing, esp. data augmentation to increase models' learning efficiency, and initial analysis. The paper researches the appliance of various techniques for the discretization of visual data (convolution, blurring, rectified linear unit, etc.) to increase the accuracy of deep learning models. As a result, the work demonstrates the appliance of a complex densely connected convolutional neural network (DenseNet) to detect anomalies (defects) in the images of nature. In order to retrieve more comprehensive result, which is not biased to some data features, there have been implemented 2 other models - EfficientNet and EfficientNet NoisyStudent. In the end, the paper presents the final result, which is an ensemble of these 3 models, with an accuracy rate of 0.965. Finally, the article gives recommendations for further research and development to improve models of anomaly detection in visual data.

Keywords [1]

computer vision, anomalies detection, data visualization, classification, deep learning

1. Introduction

In today's digital and robotic world, data analysis and computer vision technologies play a special role. Thanks to various devices for collecting visual information (cameras), it is possible to form a complete and deep picture of the surrounding environment. In fact, cameras combined with computer vision technologies, which are the development of deep learning concepts, are able to replace biological vision, and even see what is invisible to the normal eye. Actually, the detection of such features, deviations from the normal state, can significantly simplify the functioning of many spheres of human activity. Taking it into the account, the detection of features can be interpreted as one of the applied varieties of the more general task of detecting anomalies, which itself is one of the types of data classification problem [1]. There is a large number of studies on object recognition in the literature. This is especially relevant for the tasks of recognizing objects in natural conditions, since the color of the main object usually matches (or is close to) the color of the background or its elements. Agriculture

Information Technology and Implementation (IT&I-2022), November 30 - December 02, 2022, Kyiv, Ukraine
E-MAIL: yuliya.khlevna@gmail.com (Iulia Khlevna); bohkoval@gmail.com (Bohdan Koval)
ORCID: 0000-0002-1807-8450 (Iulia Khlevna); 0000-0002-3757-0221 (Bohdan Koval)

is one of the largest branches of industry where precise methods and computers increase efficiency and profitability by increasing the quality of the crop and reducing operating costs. [2]. Input data is visual data, for example, images from cameras, and drones. Such data is completely understandable to a regular user of a personal computer and is stored in common formats such as JPG, PNG or JPEG [3]. The images consist of pixels - blocks, that form NxM matrix. Therefore, it can be established that the digital representation of the image is a matrix, and in the case of a black and white image, the matrix is 2-dimensional, because each pixel can contain only one scalar value - from 0 to 255 [4].

If a black-and-white image has only 1 channel, a color image has 3 - a red channel, a green channel, and a blue channel. This representation is known as RGB format.

Thus, a color image can be represented in a digital form as a matrix of dimension NxMx3, which is a combination of 3 matrices of each of the channels: red channel, green and blue. This is an important feature because the channels are to be investigated separately, and one channel may show features that the other two do not. The paper [4] shows how to obtain information from these data: features of a particular image; what objects (contours) are on it. It was established that the quality of object recognition depends on color and background. The work [5] presents the transformation of a simple 3-dimensional matrix into a meaningful object with characteristics and attributes, which can be used when building a model. The problem of detection of three-dimensional (3D) cocircular edges defined by binocular disparity. Usually, in order to remove noise and unnecessary details, the contours of the object in the frame are highlighted. The question of the advantages and disadvantages of the background subtraction and optical flow approaches, which try to obtain a mask of mostly moving objects, as well as the methods and algorithms for determining these objects, is presented in [6]. The disadvantage is that the work presents algorithms that require additional binarization. To eliminate this, it is suggested to remove noise and unnecessary details and highlight the edges of the object (in natural conditions) in the frame.

One of the algorithms for detecting edges and structuring elements in an image is the Canny algorithm. The Canny algorithm is a technique for extracting useful structural information from various graphic objects and significantly reducing the amount of data for processing. It is widely used in various computer vision systems [3, 7]. Therefore, it is appropriate to investigate its functionality in objects that reflect natural conditions. Considering the above, it is reasonable to investigate the theory of computer vision to detect anomalies in images reflecting nature. It is appropriate to implement the application of the research in agriculture and determine the prospects of such implementation. The purpose of the work is to develop a theoretical basis for detecting anomalies in visual data of nature.

In accordance with the goal, the following research tasks were formed:

1. to investigate the methodological basis of computer vision for detecting anomalies in images of nature;
2. to develop a classifier model for detecting anomalies in images depicting nature;
3. to present the applied implementation of the model for detecting anomalies in images of the agricultural sector.

2. Computer vision methodology for anomaly detection in images

For objects that reflect natural conditions, we will use the following criteria for determining edges:

1. Edge detection with a low error rate, which means that the detection should accurately capture as many edges as possible shown in objects that reflect natural conditions.
2. The edge point determined by the algorithm must be precisely located in the center of the edge.
3. The edge in an image that represents natural conditions should be labeled only once, and where possible, image noise should not create false edges.

The work [8] presents the satisfaction of the presented requirements with the help of calculus of variations - a technique that finds a function that optimizes the sought linear function. The optimal function in the Canny detector is described by the sum of four exponential terms, but it can be approximated by the first Gaussian derivative.

Let's apply the formalized algorithm for detecting the Canny edges to objects that reflect natural conditions:

1. Applying a Gaussian filter to smooth the image that reflects natural conditions and remove noise;
2. Finding the intensity gradients of the image, which reflects natural conditions, is the key stage at which the edges of the image are revealed;
3. Applying a gradient magnitude threshold or suppressing the lower cutoff limit to get rid of the false response to edge detection of an image that reflects natural conditions;
4. Application of a double threshold to determine potential edges of an image that reflects natural conditions;
5. Edge tracking by hysteresis: finish edge detection by suppressing all other edges that are weak and not related to strong edges.

The intensity gradient of a single element (pixel) - G, and the direction of the image of the pixel - is calculated as [9] :

$$G = \sqrt{G_x^2 + G_y^2} \quad (1)$$

$$\theta = tan^{-1}(G_x^2 + G_y^2) \quad (2)$$

where G_x- the first derivative in the horizontal direction, G_y- the first derivative in the vertical direction.

The result of these five steps is a two-dimensional binary map (0 or 255) that indicates the location of the edges in the image.

In addition to the issue of algorithm selection, the relevance of data is relevant.

One of the problems generally encountered when building machine learning models is insufficient data. When trained on a limited dataset, the model may become biased and not detect all features on the test data. The reverse problem is data redundancy, which makes the model prone to overtraining. And therefore, among other things, it is necessary to correctly approach the selection and construction of the training dataset [10]. It is important to understand and use data augmentation techniques - increasing the number of dataset elements on the basis of existing ones without loss of quality. By applying 1 technique, you can double the data set. We offer to apply such techniques for visual images in natural conditions as: flipping, convolution, and blurring [11].

The selection of algorithms for training a computer vision model is of great importance. Each core ImageNet model has a different architecture, but they have common building blocks: Conv2D, MaxPool, ReLU [12]. The application of max pooling (MaxPool) is proposed in [13].

Max pooling (MaxPool) divides the input image into a set of non-overlapping rectangles, and for each such subregion, it outputs its maximum. The idea is that the exact position of a feature is not as important as its rough position relative to other features. The aggregation layer serves to gradually reduce the spatial size of the representation to reduce the number of parameters and the amount of computation in the network, and thus also to control overtraining. It is a type of nonlinear downsampling. The max pooling algorithm is very similar to convolution, except that it involves finding the maximum value in a frame instead of finding the scalar product of the frame with the kernel. The process is important to reduce the complexity of the CNN while preserving features.

ReLU is an activation function commonly used in neural network architectures. ReLU(x) returns 0 for $x < 0$ and x otherwise. This function helps introduce nonlinearity into the neural network, thus increasing its ability to model image data. ReLU is the most popular transfer function for deep neural networks [14]. For modeling computer vision to detect anomalies in images, it is suggested to use such models as DenseNet, EfficientNet, and EfficientNet Noisy Student.

In a standard convolutional neural network, we have an input image that is then passed through the network to get a predicted output label in such a way that the forward function is quite simple - linear with only a dependency on the previous layer. Each convolutional layer except the first one (which takes the input image) takes the output of the previous convolutional layer and creates an output feature map that is then passed to the next convolutional layer.

In the DenseNet architecture, each layer is connected to every other layer [15]. There are $L(L+1)/2$ direct connections for L layers. For each feature map layer, all previous layers are used as input, and each subsequent layer's own feature maps are used as input.

DenseNet is a basic, reference model that is widely used to solve computer vision problems. DenseNet introduces direct connections between any two layers with the same feature map size. The input of a layer in DenseNet is a concatenation of feature maps from previous layers. DenseNets have

several compelling advantages: they eliminate the vanishing gradient problem, enhance feature propagation, encourage feature reuse, and significantly reduce the number of parameters.

EfficientNet [16] is a convolutional neural network architecture and scaling method that uniformly scales all depth/width/resolution dimensions using a composite factor. Unlike common practice, which scales these factors arbitrarily, the EfficientNet scaling method uniformly scales the width, depth, and resolution of the network using a set of fixed scaling factors [17].

EfficientNet uses a technique called compound factor to scale models in a simple but effective way. Instead of randomly increasing width, depth, or resolution, compound scaling scales each dimension uniformly with a fixed set of scaling factors. Using the scaling method and AutoML, seven models of different dimensionality are developed that outperform the state-of-the-art accuracy of most convolutional neural networks and have much better performance.

EfficientNet Noisy Student is a separate sub-implementation of the EfficientNet model, to which the technique of semi-supervised learning using the Noisy Student method is also applied. The working algorithm of the Noisy Student method consists of 4 steps:

1. Training a classifier on labeled data (teacher).
2. Output labels on a much larger unlabeled data set.
3. Training a larger classifier on the combined set by adding noise (hence the name - noisy student).
4. Moving on to step 2, where the student acts as the teacher.

The EfficientNet model acts as a classifier.

3. Applied anomaly detection using computer vision methodology to spot plant images with defects

3.1. Initial exploratory data analysis

To develop a technology for detecting anomalies in images that represent natural conditions, a dataset of 1820 images of apple leaves was used, some of them are completely healthy, while others have certain defects, i.e. deviations from the norm - anomalies. That is, we have 3 types of defects that characterize anomalies: "scab", "corrosion", "multiple diseases". Solving this problem is important because early diagnosis of plant diseases can save tons of agricultural products annually. This will benefit not only the population as a whole by reducing hunger, but also farmers by ensuring harvests and stability of their businesses. In this work, we will try to distinguish healthy (normal) plant leaves from anomaly (defective) ones.

The implemented solution is based on the Python programming language ecosystem using the following helper libraries:

- *pandas* - a library for data processing and analysis. Offers data structures and operations for working with numeric tables and time series.
- *numpy* and *scipy* - add support for large multidimensional arrays and matrices, along with a large collection of high-level math functions for working with these arrays.
- *keras* and *tensorflow* - libraries for machine learning and artificial intelligence. They can be used in a number of tasks, but specialize in training deep neural networks.
- *seaborn*, *matplotlib* and *plotly* - visualization libraries.
- *OpenCV* - a library of functions aimed primarily at real-time computer vision.

The dataset consists of 3 input data types:

- *train_data.csv* - a table with training, labeled data. We have a link to the image and its classification (whether it is healthy or has certain defects - corrosion, scabies or multiple diseases).
- *image* - JPG images of the leaves referenced by the test dataset. In the upcoming section, we will show examples of healthy and defective leaves.
- *test_data.csv* - a table for testing, which also has a reference to the image, but this data is not labeled. Actually classifying these images is our task.

With the help of the OpenCV (Open Source Computer Vision) library, we will read examples of each of the 4 possible variations of the object, which are presented in Fig. 1.

From Fig. 1, it was established that the problem is not only the detection of the anomaly as such (the presence of a defect), but also the classification of this defect. Moreover, the task is complicated by the fact that a separate group (several diseases) is a composite group of two others - corrosion and scabies (which already have many common features).

Figure 1: Example of input images: top left - a healthy leaf, top right - a leaf with scab, bottom left - a leaf with corrosion, bottom right - a leaf with several diseases (usually both scab and corrosion).

The output images will be received by the model in the form of 3 channels - red, green, and blue. Fig. 2 provides an example of layers (channels) matrix visualization and color distribution.

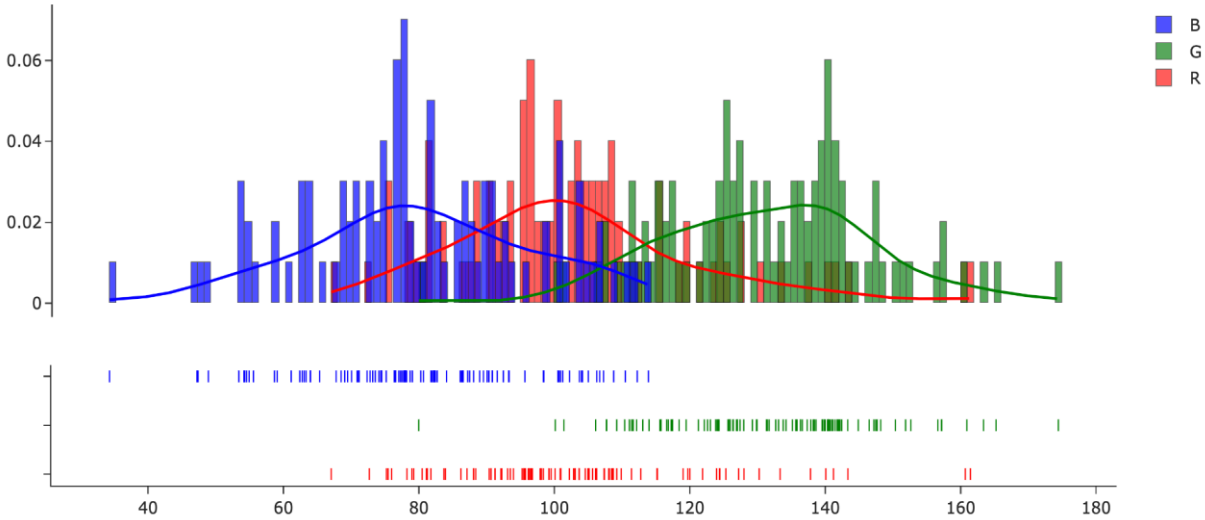

Figure 2: An example of the layers (channels) matrix visualization on the example of image 3 - a healthy leaf.

In general, from 1820 images, the classes are distributed as follows:

- healthy - 28.3%
- with scab - 32.5%
- with corrosion - 34.2%
- with multiple diseases - 5%.

This distribution greatly simplifies the work, because usually anomalies occur less often than in normal cases (there are usually more healthy leaves than unhealthy ones, even possibly on an unhealthy tree). Therefore, at the very least, we do not need to apply discretization or data sampling methods to achieve an approximately equal ratio - the dataset already satisfies this requirement. The "multiple

diseases" category has a much smaller percentage, but it is a combination of the other two, and therefore we can assign the image to this category if we see a defect on it, but we cannot determine which one.

3.2. Data preprocessing prior to computer vision modeling

We will apply the Canny algorithm from the OpenCV library (cv2.Canny). The result is presented on Fig. 3 below.

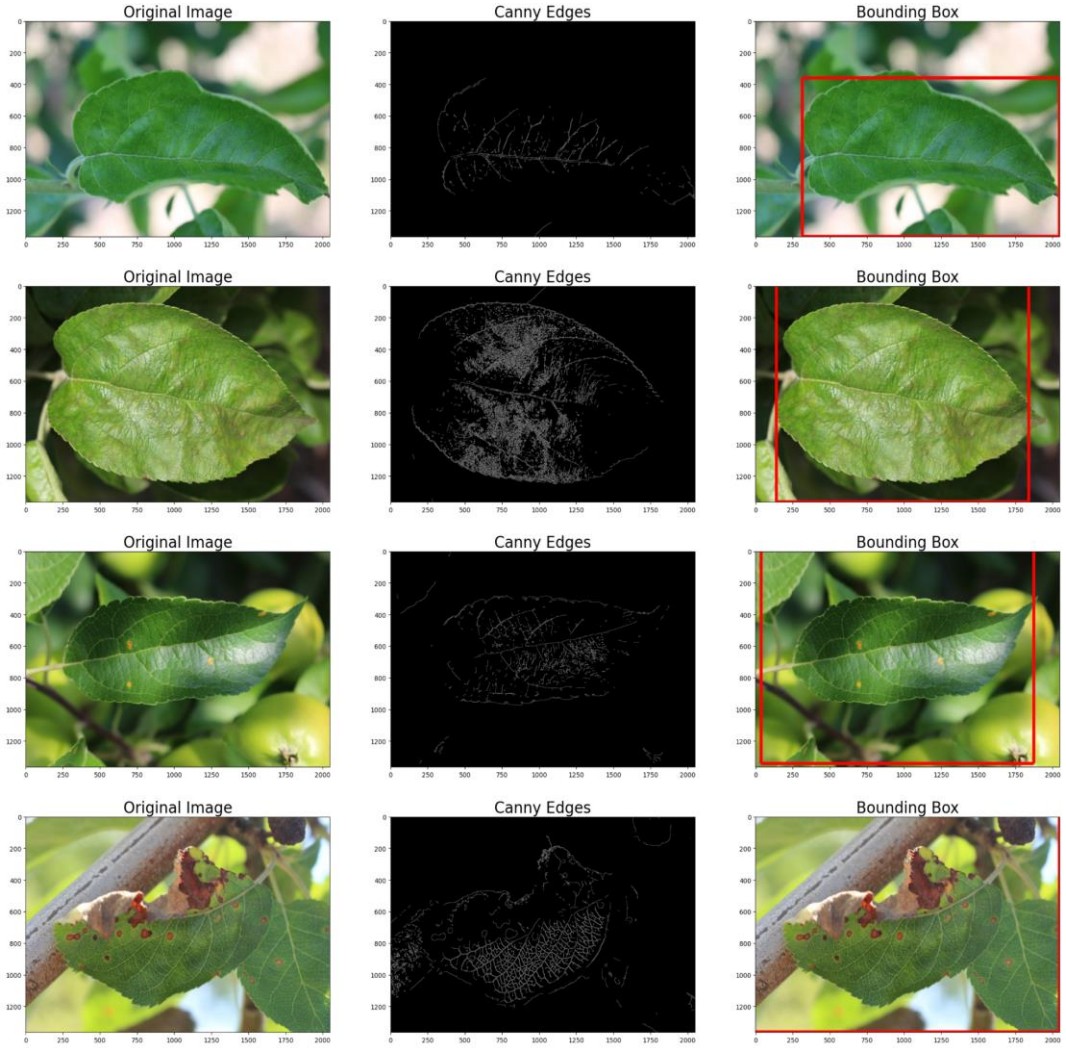

Figure 3: The result of the Canny algorithm on each of the leaf types

In practice, the Canny algorithm is immediately useful - it allows us to isolate the edges of the element, and thus reduce the dimensionality of the matrix and speed up processing.

Next, we will apply data augmentation techniques to expand our dataset, namely flipping, convolution, and blurring, described in the previous section. Let's display the result for a separate leaf to clearly demonstrate the result of the flipping algorithm (Fig. 4), the convolution algorithm (Fig. 5), and the blurring algorithm (Fig. 6).

3.3. Computer vision models to detect anomalies on plant images

We will use the keras package to build the DenseNet model (Fig. 7). In Fig. 7 we can see the results of model training, from which we can see that the model reached the plateau of results at 20 iterations, and the accuracy of the model on test data was 0.9525548. In Fig. 8 we see an example of the classification of an individual leaf by the DenseNet network.

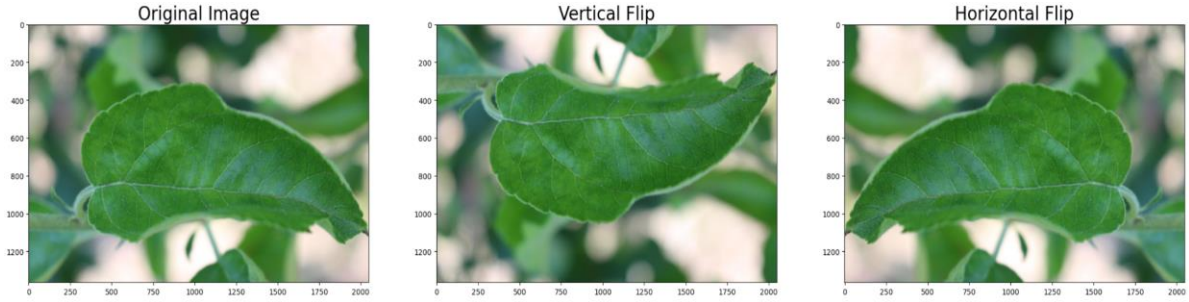

Figure 4: The result of the flipping algorithm (horizontal and vertical)

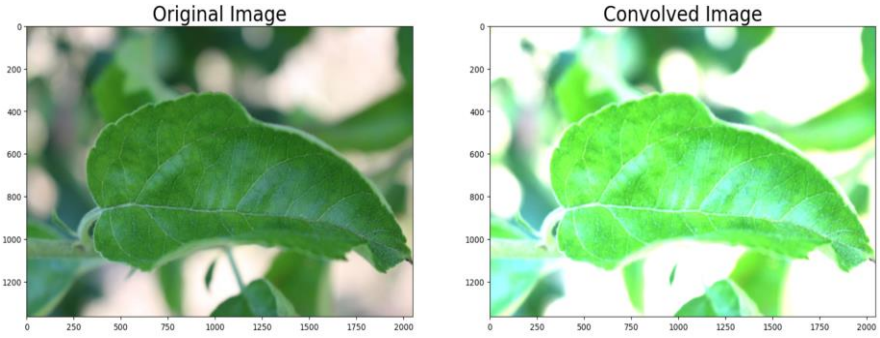

Figure 5: The result of the convolution algorithm

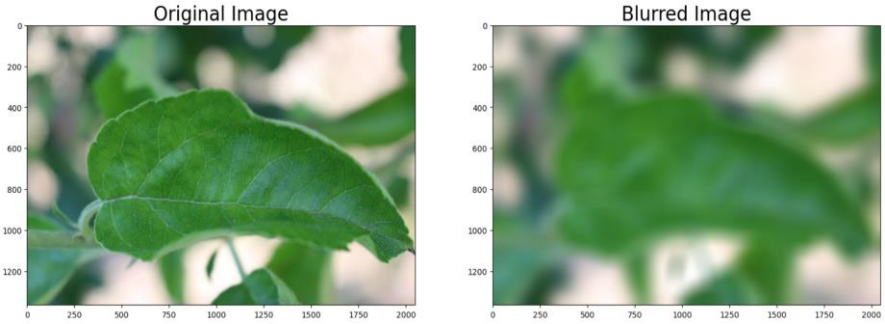

Figure 6: The result of the blurring algorithm

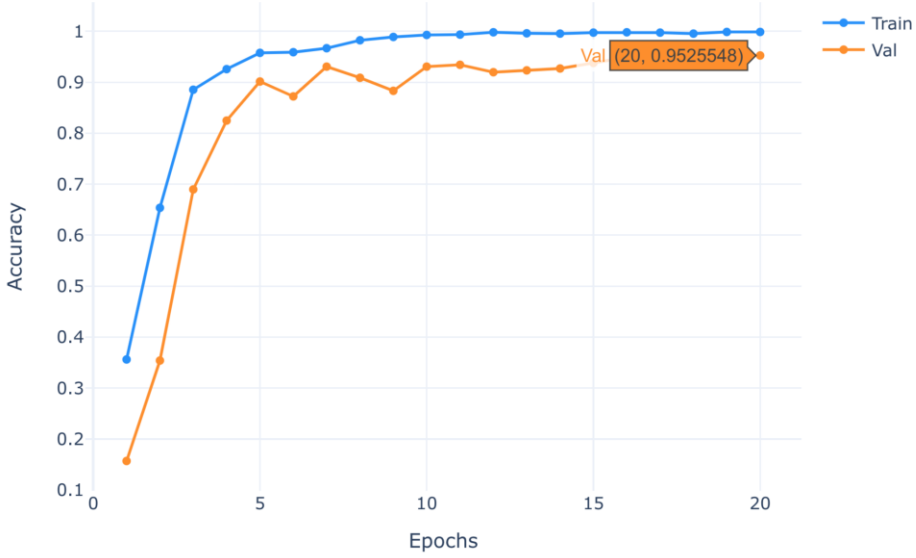

Figure 7: The result of neural network training based on DenseNet

Therefore, the model classified the leaf as "multiple diseases", which is the correct classification. The model swayed very slightly towards “corrosion” (and indeed, there are brown dots on the leaf that characterize corrosion), but with considerable confidence, DenseNet classified the leaf as “multiple

diseases" because it clearly shows significant defects of both corrosion and scab. Next, we initialize the network based on the EfficientNet model. Fig. 9 shows the results of the model training. It was established that the indicators also stabilized at 20 iterations (20 iterations are, in principle, the optimal number for most neural networks). The accuracy of the model was 0.9452555 on the test data. The fundamental blocks are shown in Fig. 10.

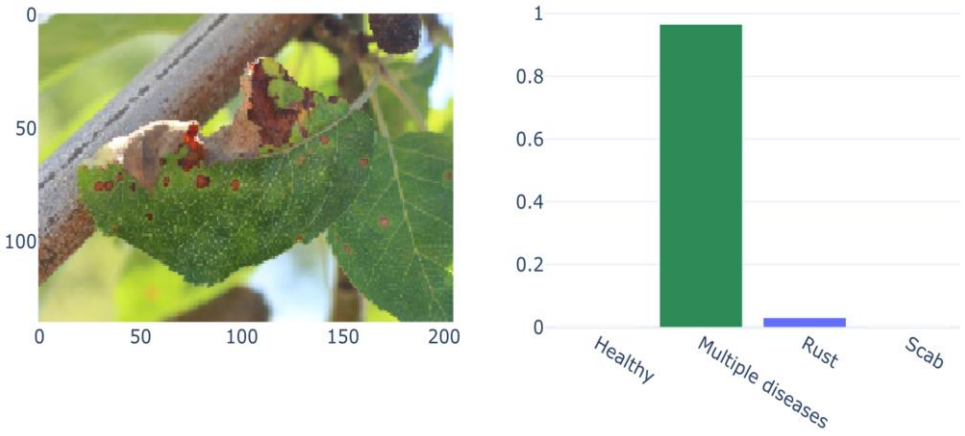

Figure 8: The result of the classification of a single leaf by the DenseNet network.

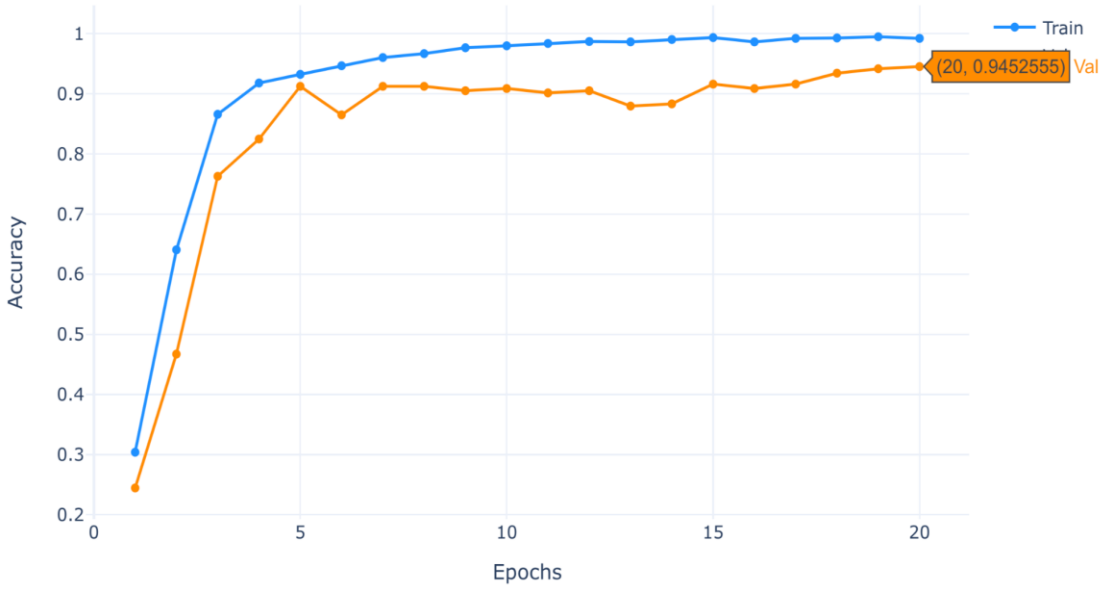

Figure 9: The result of neural network training based on EfficientNet

The EfficientNet network processes results similarly to DenseNet. The final classification contains weights from 0 to 1 for each class, from which EfficientNet chooses the largest. Similarly, we initialize the EfficientNet Noisy Student model. Similarly to DenseNet and EfficientNet, the fundamental blocks are also shown in Fig. 10. The implementation of EfficientNet Noisy Student is fundamentally similar to EfficientNet, because, in fact, it only has an extension in the form of Noisy Student over the classic EfficientNet. According to the results of training of the EfficientNet Noisy Student model, which showed an accuracy of 0.9160584 (Fig. 11), it can be concluded that, in general, the Noisy Student add-on worsened the result, but the data obtained from the research was not excluded.

So, the results of the developed models are:

1. DenseNet - 0.9525548
2. EfficientNet - 0.9452555

EfficientNet Noisy Student - 0.9160584

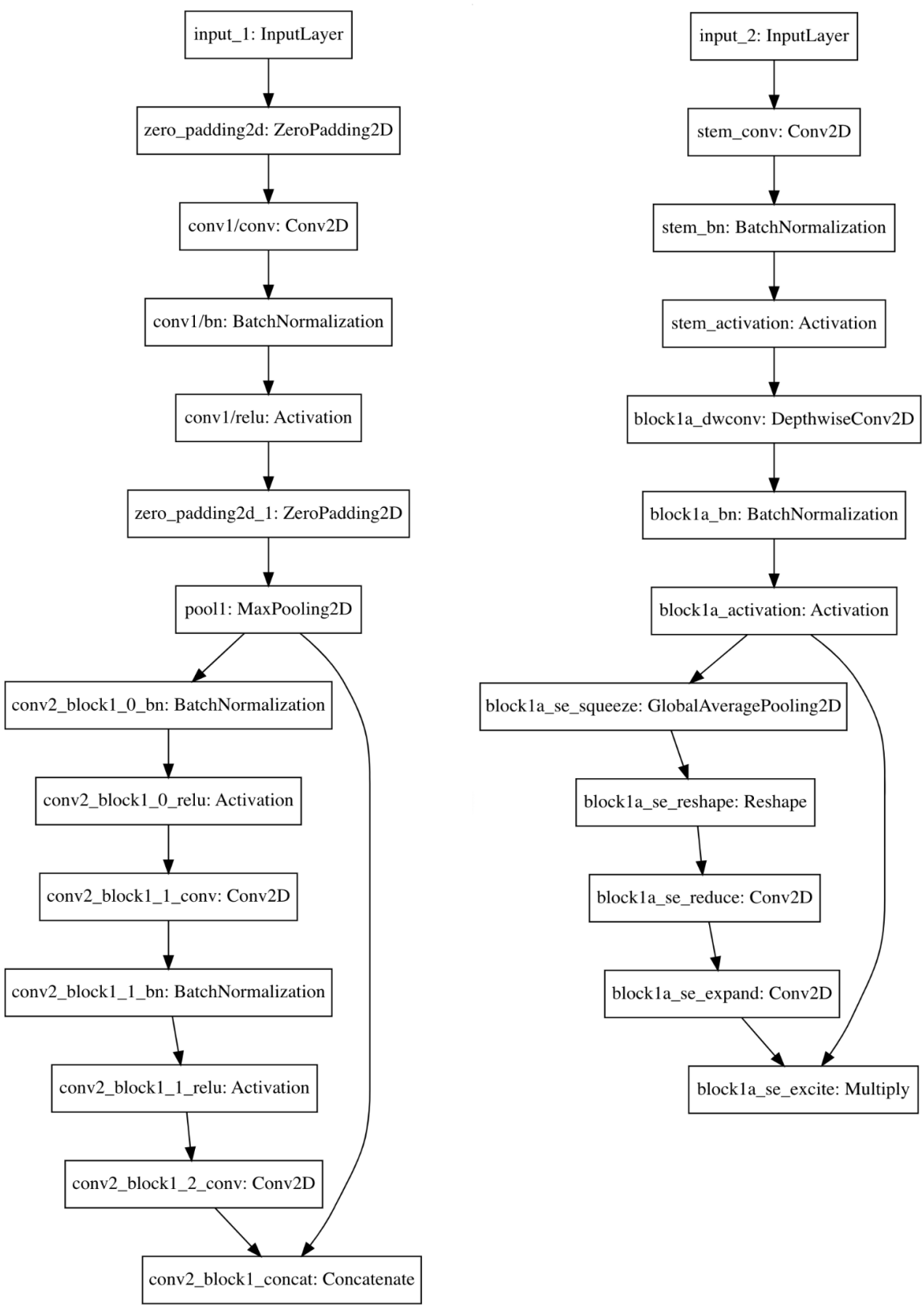

Figure 10: The fundamental blocks of the DenseNet (left) and EfficientNet (right) models

It would be possible to end the work there and define DenseNet as the most accurate and apply it, but we have another powerful tool in our arsenal - model ensembles. The final result will be determined at once with the help of all 3 models - they will vote for each of the images. The classification of the image that receives the most votes (3 or 2) is considered the final result of the calculation. If parity is

established (classifications have one vote each), then DenseNet is preferred as the most accurate. But, if both EfficientNet and EfficientNet Noisy Student convince us that the DenseNet classification is not correct - we prefer the majority vote. As a result of the strategy chosen above, we get an ensemble result with an accuracy of 0.965. In general, it can be said that the ensemble of models was beneficial.

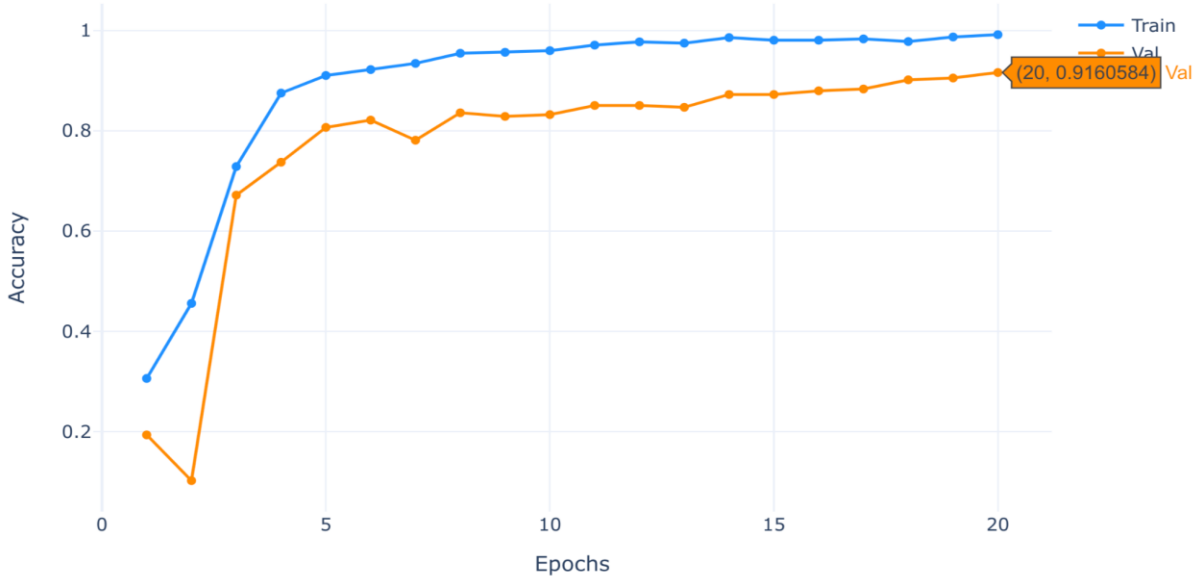

Figure 11: The result of neural network training based on EfficientNet Noisy Student

3. Conclusion and perspectives

The paper is dedicated to solving the problem of detecting anomalies in visual data using computer vision technology. The task of anomaly detection is considered as a subtype of the classification problem using deep learning. Visual data (images) were processed as 3-dimensional matrices, each of the 3 dimensions corresponded to one of the color channels - red, green and blue, which is the basis of the digital representation of the image in the RBG format. The work provides algorithms for working with visual data, including the Canny algorithm for edge detection, as well as flipping, convolution, and blurring algorithms for data augmentation. The applied models of neural networks are described: DenseNet, EfficientNet and EfficientNet Noisy Student, which are examples of convolutional neural networks and are suitable for solving classification problems on visual data. Also, the article mentions the constituent elements of these networks, such as max pooling and the activation function based on a rectified linear unit (ReLU).

3 neural networks were implemented based on the mentioned models, as well as their ensemble, which became the final decision-making algorithm. If we evaluate the models separately, DenseNet showed the best result with an accuracy of 0.9525548, EfficientNet was slightly behind it with a result of 0.9452555, and EfficientNet Noisy Student came close with an accuracy of 0.9160584. The ensemble of models showed a result with an accuracy of 0.965, and therefore it can be argued that it is a powerful tool for improving the efficiency of models. The implemented concept of detecting anomalies (defects) on plant leaves demonstrated its feasibility. Existing models show results within 0.95 (which is actually reflected in our work), and therefore the ensemble of models may contain the key to further improvement of models.

Finally, we will provide recommendations that can potentially improve the result:

1. Build a more complex ensemble of models to take into account the weights determined by each individual model. Thus, operate not only with Boolean expressions (0 or 1), but approach more comprehensively to voting models taking into account weights. Potentially it is possible that the model will not have 1 vote, but, for example, 1000, and can give a part for class 1, another for class 2, etc. This will help to approach the ensemble of models in a more complex way.
2. Modification of neural network model architectures. We used the classical implementation of DenseNet and EfficientNet. Alternatively, we can customize them: the number of layers, interactions

between them, etc. The TensorFlow library used is fairly standardized, so we can consider PyTorch as an alternative that offers greater model flexibility.

5. References

[1] Khlevna I, Koval B (2020) Fraud detection technology in payment systems. In: IT&I 2020—Information technology and interactions. Proceedings of the 7th international conference "information technology and interactions" (IT&I-2020). Workshops proceedings, Kyiv, Ukraine, 2–3 Dec, 2020. CEUR Workshop Proceedings, pp 85–95.

[2] Kamilaris, Andreas, and Francesc X. Prenafeta-Boldú. A review of the use of convolutional neural networks in agriculture. The Journal of Agricultural Science 156.3 (2018): 312-322. URL:https://proceedings.neurips.cc/paper/2015/file/536a76f94cf7535158f66cfbd4b113b6-Paper.pdf

[3] Ruder, Sebastian. "An overview of gradient descent optimization algorithms." arXiv:1609.04747 (2016). URL: https://arxiv.org/pdf/1609.04747.pdf

[4] N. Kussul, M. Lavreniuk, S. Skakun and A. Shelestov, "Deep Learning Classification of Land Cover and Crop Types Using Remote Sensing Data," in IEEE Geoscience and Remote Sensing Letters, vol. 14, no. 5, pp. 778-782, May 2017, doi: 10.1109/LGRS.2017.2681128.

[5] Sieu K. Khuu, Vanessa Honson, Juno Kim; The perception of three-dimensional contours and the effect of luminance polarity and color change on their detection. *Journal of Vision* 2016;16(3):31. doi: https://doi.org/10.1167/16.3.31.

[6] Khlevna, I., Zhovtukhin, D. Using Image Segmentation Neural Network Model for Motion Representation in Sport Analytics. Lecture Notes in Networks and Systems, 2022, 344, pp. 363–375

[7] Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. Deep learning. MIT press, 2016.

[8] Traore, Boukaye Boubacar, Bernard Kamsu-Foguem, and Fana Tangara. "Deep convolution neural network for image recognition." Ecological Informatics 48 (2018): 257-268.

[9] Canny J. A computational approach to edge detection. IEEE Transactions on pattern analysis and machine intelligence. 1986 Nov(6):679-98.

[10] Van der Walt, Stefan, Johannes L. Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D. Warner, Neil Yager, Emmanuelle Gouillart, and Tony Yu. "scikit-image: image processing in Python."PeerJ 2 (2014): e453

[11] Matteoli, Stefania, Marco Diani, and Giovanni Corsini. "A tutorial overview of anomaly detection in hyperspectral images." IEEE Aerospace and Electronic Systems Magazine 25.7 (2010): 5-28.

[12] Ciolino, Matthew & Noever, D. & Kalin, Josh. (2019). Training Set Affect on Super Resolution for Automated Target Recognition. URL: https://www.researchgate.net/publication/337386697_Training_Set_Affect_on_Super_Resolution_for_Automated_Target_Recognition

[13] K. Simonyan, A. Zisserman. Very Deep Convolutional Networks for LargeScale Image Recognition., квіт 2015, URL: http://arxiv.org/abs/1409.1556.

[14] Romanuke, Vadim. "Appropriate number and allocation of ReLUs in convolutional neural networks." Research Bulletin of the National Technical University of Ukraine" Kyiv Politechnic Institute" 1 (2017): 69-78.

[15] G. Huang, Z. Liu, K. Q. Weinberger, and L. Maaten. Densely connected convolutional networks. In CVPR, 2017. URL: https://arxiv.org/abs/1608.06993

[16] Koonce,B. (2021). EfficientNet. In: Convolutional Neural Networks with Swift for Tensorflow. Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-6168-2_10

[17] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. ICML, 2019. URL: https://arxiv.org/abs/1905.11946